Large scale continuous integration and delivery

Stahl, Daniel

Chapter 8. Industry Application of Continuous Integration Modeling: A Multiple-Case Study

This chapter is published as:
Ståhl, D., & Bosch, J. (2016). Industry application of continuous integration modeling: a multiple-case study. In *Proceedings of the 38th International Conference on Software Engineering Companion* (pp. 270-279). ACM.

Abstract
The popular agile practice of continuous integration has become an essential part of the software development process in many companies, sometimes to the extent that delivery to customer is impossible without it. Due to this pivotal role it is an important field of research to better understand the practice: continuous integration system behavior, improvement identification and analysis of change impacts. This paper investigates the effects of modeling of such systems, by applying two continuous integration modeling techniques to four separate industry cases in three companies. These techniques are found to complement each other, and their ability to help professionals gain a better understanding of their continuous integration systems, to communicate around them and to support technical work is demonstrated. In addition, guidelines for conducting similar continuous integration modeling work are presented and feedback on the studied models provided. This work presents software professionals with demonstrably effective methods for design and analysis of continuous integration systems and thereby improving the efficacy of a vital part of their software development efforts, while supporting researchers with recommendations for and feedback on available modeling techniques.

8.1 Introduction

While the agile practice of continuous integration has received considerable attention in the last decade – not least in industry, where the employment of, or at least aspiration to continuous integration is prevalent – the precise nature of continuous integration implementations are typically superficially and ambiguously described [Ståhl 2014a]. At the same time, continuous integration (and delivery) is known to be complicated and difficult to get right, particularly in enterprise scale development [Roberts 2004, Rogers 2004] where continuous integration systems (the coherent system of infrastructure, tools and processes applied to achieved the practice) can be highly complex and require dozens or hundreds of engineers to develop and maintain – indeed, it should be no surprise that the systems that integrate, build and test complex very-large-scale software systems are themselves highly complex. We also find that these systems tend to yield varying results [Ståhl 2013]. It is furthermore difficult to obtain an end-to-end overview and understanding of such a large scale continuous integration system and the effects of integration results [Alyahya 2011], not least since large numbers of people in multiple roles [Krusche 2014], as well as a multitude of tools, such as build tools, version control systems and test automation frameworks, are involved [Hoffman 2009, Kim 2009a, Yuksel 2009], all while operating under strict time constraints [Jiang 2012].

It follows then that it can be difficult both to understand and to agree on where and how to improve a continuous integration system. At the same time, continuous integration has evolved into an integral and business critical element of software development in many companies, and an area

of enormous investment: in our work we frequently encounter companies and projects where tens of millions of dollars are invested, where organizations of hundreds of engineers are dedicated to supporting continuous integration, and where as much as 25% of the total development capacity is tied up building and maintaining continuous integration and delivery capabilities. Consequently, we consider it an important area of research to find methods supporting practitioners in achieving a holistic view of their continuous integration systems – better understanding them, determining how to improve them and how to predict the implications of such improvement efforts.

As it has long been generally accepted that models and diagrams can both help understand and communicate complicated software systems, we posit that modeling holds great promise for addressing these problems in the domain of continuous integration. In previous work we have participated in the development of two different modeling techniques – Automated Software Integration Flows (ASIF) [Ståhl 2014a, Ståhl 2014b] and Continuous Integration Visualization Technique (CIViT) [Nilsson 2014] – created in different contexts, driven by different needs and serving different purposes; yet they both describe the same phenomenon. Consequently we have investigated the industry application of these techniques in tandem, specifically analyzing their ability to complement one another and their ability to bring benefit to industry practitioners in their work. Based on our findings we also present methods for effective continuous integration modeling, and discuss our results and the studied modeling techniques in the context of related work within the research community.

The contribution of this paper is three-fold. First, it demonstrates that modeling is an effective practice in analyzing, designing and communicating continuous integration systems. Second, it shows that available modeling techniques specifically designed for the continuous integration domain can effectively be used as complements to one another. Third, it affirms in vivo applicability of these modeling techniques to large scale industrial cases and provides practitioners and researchers with recommendations for their use.

The remainder of this paper is structured as follows. The next section provides a background and introduction to continuous integration modeling techniques, followed by our research questions. Then, in Section 8.3, the research method is presented. Section 8.4 presents the results of the multiple-case study and the subsequent analysis. Related work is then reviewed in Section 8.5 and threats to validity are discussed in Section 8.6, whereupon the paper is concluded in Section 8.7.

8.2 Background

In previous work we have established that not only are there divergent expectations on the practice of continuous integration in industry [Ståhl 2013], but its implementation also varies greatly [Ståhl 2014a]. Specifically to address the detected variation points of the practice, a modeling technique, Automated Software Integration Flow (ASIF) was developed, evolved and successfully applied to multiple large-scale industry projects and used to phrase guidelines for continuous integration implementation [Ståhl 2014b].

In parallel a separate and independent modeling technique for continuous integration – Continuous Integration Visualization Technique (CIViT) – was developed and also successfully used in industry settings [Nilsson 2014], with one of us participating in the original research.

While both techniques, used independently, are demonstrably effective, we posit that as they ultimately address the same problem domain and strive to enhance understanding of the same practice, they should also be compatible in the sense that they can favorably be used to offer complementary views, possibly satisfying different needs from different perspectives.

The remainder of this section describes the two modeling techniques in greater detail, compares them, and presents the research questions motivating the work reported from in this paper.

8.2.1 Modeling Techniques Introduction

Modeling is fundamental to and used to great effect in many aspects of software engineering. The positive results in previous work described above notwithstanding, the use of modeling to describe, analyze and communicate continuous integration systems is in our experience uncommon in the industry, and arguably under-researched – a research area not to be confused with the continuous integration of model based software. As discussed further in Section 8.5, we have searched for modeling and visualization approaches to the design of continuous integration systems, but to our great surprise only found work on two techniques, which we ourselves have been involved in. Consequently, we have proceeded by including these two techniques in our study.

8.2.1.1 The CIViT Technique

The Continuous Integration Visualization Technique (CIViT) was developed through a multiple-case study – in which one of the authors of this paper participated – of large companies "striving towards continuous deployment of software", with the aim of visualizing "end-to-end testing activities in order to support the transformation towards continuous integration" [Nilsson 2014]. In CIViT, tests are split into four categories: testing new functionality requirements, legacy functionality requirements, quality requirements and edge cases (defined as "unlikely or weird situations", often discovered through considerable investigative effort in customer systems): F, L, Q and E, respectively. The test coverage in each category is measured along with the degree of automation and the feedback loop length at various test scopes (e.g. at "component" level and "full product" level), all of which is plotted in a graph, as exemplified in Figure 32.

8.2.1.2 The ASIF Technique

The Automated Software Integration Flow (ASIF) technique offers a graphical view of an integration system, where nodes represent activities, inputs and triggering factors, and edges show input consuming relationships and triggering relationships, respectively. An example instantiation is shown in Figure 33. The modeling technique particularly focuses on the automated activities and their characteristics, building on the concept of software integration as a Directed Acyclic Graph (DAG) of interconnected activities [Beaumont 2012].

Figure 32: An example of a CIViT model. As can be seen from the green icon borders there is a high degree of automation in lower system tiers, but much less so in the Release and Customer tiers. At the same time, it's worth noting that the green filling signifies that it is mostly at these higher system tiers that a high degree of test coverage is achieved.

It should be noted that the ASIF technique does not prescribe which attributes to include in the model. Depending on the choice of attributes the model may provide both qualitative and quantitative data, and it may be used to depict different aspects of the integration flow. A base set of attributes designed to cover variation points found in literature has been proposed by us in previous work [Ståhl 2014a]. In this particular study, however, we have chosen to include attributes that map as closely as possible to CIViT concepts ($Att_{1\text{-}6}$), in order to capture any cross-fertilization of the two techniques. In addition, attributes that in previous work have been found to be of great interest to professionals ($Att_{7\text{-}13}$) were included.

- Att_1: The steps required before integrating new code with the integration target. Possible values: `{none, review, queue, tests, automated}`.
- Att_2: The average duration of the activity, measured in minutes. Possible values: `{0-5, 5-30, 30-60, 60-120, 120+}`.
- $Att_{3\text{-}6}$: The percentage of requirements, explicit or implicit, as estimated by the responsible engineers, covered by (automated) tests performed by the activity. Possible values: `{0-5, 5-25, 25-75, 75-95, 95-100}`. These attributes correspond to the four test types documented in the CIViT model [Nilsson 2014].
- Att_7: The type of branch or repository the input node represents. Possible values: `{private, team, development, release}`.

- Att_8: Whether the activity involves deployment of the software, either to lab equipment or to customers. Possible values: `{none, lab, customer}`.
- Att_9: Whether the activity performs any code analysis (e.g. memory profiling, style checks or complexity analysis). Possible values: `{none, static, dynamic}`.
- Att_{10}: What constitutes a successful verdict of the activity. Possible values: `{none, non-catastrophic, tests-passed, metrics-satisfactory, artifacts-built}`.
- Att_{11}: How the status of the activity is communicated to its stakeholders. Possible values: `{none, mail, web-page, radiator-screen, reporting-meeting, other}`.
- Att_{12}: The typical interval from a failure verdict until the subsequent success verdict. This is only applied to activities that operate in a non-private context, as the information is irrelevant on e.g. a developer's private test branch. Possible values: `{n/a, minutes, hours, days}`.
- Att_{13}: The percentage of activity executions that result in a failure verdict. Possible values: `{0-1, 1-10, 10-25, 25-50, 50-100}`.

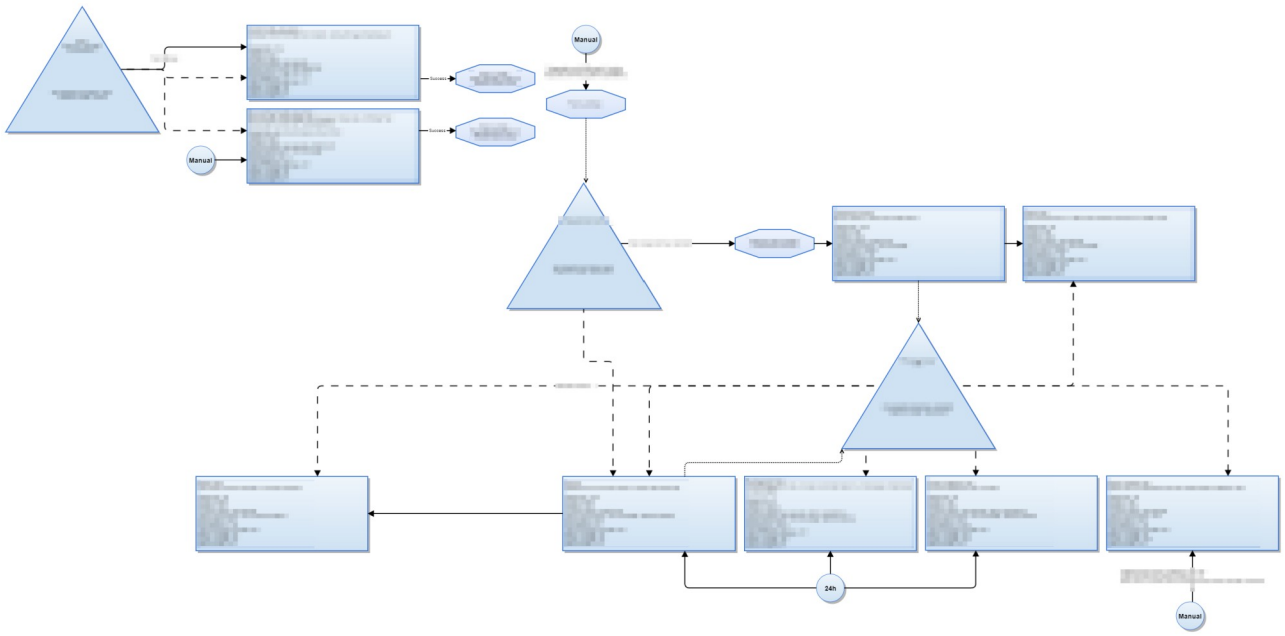

Figure 33: The ASIF model representing the current situation of one of the cases, included to provide an outline of the nature of the technique. This was the first model type to be created in each of the four cases. Triangles, rectangles, circles and hexagons represent inputs, automated activities, triggers and janitorial tasks, respectively. Textual information has been intentionally blurred.

Att_1 in combination with Att_2 and the shape of the ASIF Directed Acyclic Graph (DAG) can be used to determine the feedback loop lengths of the CIViT model. What in CIViT is referred to as test scope, e.g. component scope or product scope, is derived from the context of an activity in the ASIF graph itself.

As can be seen above, the attributes are limited to small sets of possible values. This approach of classification over qualitative descriptions or precise quantitative values in the node attributes was deemed sufficient due to the similarly rough classification used in CIViT. In addition, in previous work we have frequently experienced that engineers find it difficult to estimate quantitative attributes with any precision – therefore we consider ostensibly exact values to be at best meaningless, and quite possibly deceptive.

During the case studies reported from in this paper we used a modified variant of the technique where what we term janitorial tasks (e.g. automatic merging of source code or voting in code review tools), to which the activity attributes are not applicable, were represented as hexagon graph nodes with a short description, rather than as regular activity nodes, as shown in Figure 33.

8.2.2 Research Questions

The two techniques presented above approach the problem of software integration modeling at different levels of abstraction and address different needs. The CIViT model does not deal with the individual activities in detail, but instead focuses on overall capabilities in the integration flow: what is tested, is it automated, and when is feedback made available?

As for the ASIF technique, its detailed view of individual activities and their causal relationships contains much more information – given the right attributes, it could in fact constitute a superset of the CIViT information. Partly due to this richness, interpreting it may require some effort, however. Where high level information on how the integration ow as a whole serves its users has been obtained in previous work it has been through comparison and analysis of multiple modeled cases along with a considerable number of interviews with developers, testers and managers – a process which is slow, labor intensive and clearly impractical for industry application.

Based on this reasoning we posit that the two modeling techniques may complement each other: ASIF can provide a detailed picture of the integration system, while CIViT, viewing the same data from a different perspective, can provide a more easily accessible overview. This may aid the identification of areas of improvement, an ability we consider particularly important when aspiring to rapid release schedules, typically following on the heels of continuous integration, as there is some reason to believe that this shrinks the test scope [Mäntylä 2013] and may even adversely affect software quality [Khomh 2012]. In addition to this, it is conceivable that ASIF modeling could serve to clarify how proposed improvements could be implemented and thereby help analyze their implications.

While investigating this, we also wish to seize the opportunity of contributing any resulting experiences and insights from performing continuous integration modeling, leading us to the following research questions:

- **RQ_1:** How may continuous integration modeling be applied to benefit industry practitioners in their day-to-day work?
- **RQ_2:** How can the continuous integration specific modeling techniques of ASIF and CIViT be effectively applied in tandem to complement each other?

8.3 Research Method

In order to address the research questions a multiple-case study of self-described continuous integration projects in industry was carried out, where the studied modeling techniques could be applied and evaluated. Recognizing that expectations on, experiences from and design of continuous integration systems vary greatly [Ståhl 2013, Ståhl 2014a] we found it important to study not just a single case, but multiple, in order to achieve data triangulation [Runeson 2009].

The research was executed as a Software Center[18] research project – Software Center is a partnership between five universities and seven non-competing companies to provide a shared research and development environment. Consequently, the studied cases were selected from among the Software Center partner companies. Each case was followed up by interviews with a total of ten case study participants. The results from the interviews were then analyzed along with our own observations from the multiple-case study in order to produce our results.

In each studied case the selected modeling techniques were applied in modeling workshop sessions. Participants in these workshops were engineers of various roles (e.g. integration architect, configuration manager, project manager, enterprise architect and test coordinator), appointed by their respective companies as knowledgeable of the integration systems. Based on our previous experiences from similar work [Ståhl 2014b] our requirements were that the very engineers responsible for building and maintaining the software integration systems be present, as they are most likely to possess the required level of detailed knowledge of the systems.

The models were created – with the researcher at the keyboard – in real time during these workshops and shown on a big screen or overhead projector, enabling all participants to verify their accuracy as they were being assembled. Due to logistical necessities the workshops of all cases were split into multiple sessions, some of them conducted over video conference, with a total effective time spent varying between two and a half and seven hours. Each case contained at least one face-to-face session.

The modeling sessions were designed to address the research questions, based on our reasoning with regards to the respective strengths and weaknesses of the modeling techniques and their potential to complement one another (see Section 8.2.2): the first step in each session was to create an ASIF model describing the current state. Then, where possible using the data available from the ASIF model, a CIViT model was created. Finally the ASIF model was manipulated to represent a wanted state.

Following the modeling sessions, interviews were conducted with the participants. To achieve data and observer triangulation [Runeson 2009] as many interviewees of the varied participant roles as possible were included, reaching a total of ten engineers across all four cases. The minority of participants who were not interviewed were excluded because they only participated during parts of the sessions, or because they were unavailable for interviews.

18 http://www.software-center.se

To what extent do you consider...	
IQ_1	... the Current ASIF helpful in defining the automated activities of the CIViT model?
IQ_2	... the CIViT model helpful in building the Wanted Position ASIF model by providing high level goals?
IQ_3	... the Wanted Position ASIF model helpful in writing requirements and/or technical specifications for integration system improvements?
IQ_4	... the CIViT and ASIF modeling techniques combined an effective means of planning integration system improvements?

Table 20: Interview questions.

Similarly to the modeling session design, the interview questions were phrased based on our research questions and reasoning with regards to the strengths and weaknesses of the techniques (see Section 8.2.2). The interviews were semi-structured and carried out individually, either face-to-face, over phone or over video conference, as circumstances permitted, with responses transcribed and read back to the interviewees during the interviews. The interview guide consisted of four opinion/value interview questions [Hove 2005] (see Table 20). The interview questions were designed to answer the research questions by obtaining the participants' experiences from and reflections on the application of the modeling techniques in the modeling workshops. Specifically, based on our reasoning regarding the respective strengths of the models, we wanted to know whether moving from one model to another was in any way helpful (IQ_1 and IQ_2), whether they found the final resulting model useful in assisting technical work (IQ_3), and whether they considered these modeling techniques a favorable approach in planning improvements (IQ_4). Furthermore, in order to capture additional aspects not foreseen by us, the interviewees were explicitly encouraged to provide their own spontaneous input and to reflect on whether there were any additional questions they thought should have been in the interview guide, but were not.

The interviewee responses and our observations from the modeling workshops were then collated and thematically coded [Robson 2011] as pertaining to effects on understanding and communicating, modeling as basis for improvement, complementary effects of the studied techniques, recommendations for continuous integration modeling practice or as feedback on the studied techniques. The resulting thematic sets were then analyzed and interpreted individually to produce the conclusions (see Section 8.4.3).

8.4 Results

Three Software Center partner companies participated in the multiple-case study:

- **Company I** is a multinational mobile phone manufacturing company, producing a wide variety of portable devices and wireless systems.

- **Company II** is a Swedish defense and aerospace company, developing both military and civilian aircraft as well as ground combat weapons, missile systems and electronic defense systems.
- **Company III** is a multinational provider of telecommunication networks, networking equipment and operator services as well as television and video systems.

From these companies, four cases were selected for inclusion in the study:

- **Case A** is the integration system for Android device software at Company I.
- **Case B** is the integration of jet fighter aircraft software systems produced by Company II.
- **Case C** is the integration of mobile device modem software at Company III.
- **Case D** is the integration system for the software of a router product at Company III.

All of the studied cases are relatively large, ranging in development project size from hundreds to thousands of personnel. They all operate in different markets under very different circumstances, restricting and in and delivery processes. To exemplify, the development of Android device software is tightly linked to the Android release process, while the development of jet fighter aircraft operates in a strictly regulated market, whose customers have very different requirements and expectations compared to users of consumer electronics.

Consequently, we believe that the selected cases from distinctly different industries represent an excellent opportunity for further validation of the models, particularly given that previous work has already shown the applicability of ASIF to projects of very different sizes [Ståhl 2014b].

8.4.1 Modeling Workshops

This section presents the results from the modeling sessions and interviews, followed by analysis. The complete data sets and models produced in the workshops are not included as they are not of primary interest in answering our research question. Instead it is the experiences gained from developing those models, and the participants' reflections on those experiences, that constitutes the relevant case data.

8.4.1.1 Case A

In the interest of time efficiency and to limit the need for travels the case study started with two one hour video conference sessions where the engineers presented their integration system and the modeling techniques were explained, respectively. Based on this we created an initial ASIF model offline, which served as input to a five hour face-to-face workshop where all the models were constructed, reviewed and discussed. All this occurred under intense debate among the participating engineers regarding their integration system, its capabilities, performance and purpose.

8.4.1.2 Case B

While creating the CIViT model of case B, the workshop participants commented that a number of the test activities performed span over several levels of both system completeness and feedback time (the vertical and horizontal axes of the CIViT diagram, respectively), but how this should be represented in the model was unclear.

It was also commented that both models are information dense. One participant, having had previous experience of CIViT, related how he with favorable results had stripped a CIViT model of all color coding representing automation and coverage, keeping only test activity labels plotted in the diagram, in order to make it presentable to an uninitiated management group.

The case study was performed in two workshop sessions, two months apart, totaling approximately six hours.

8.4.1.3 Case C

During the modeling workshop of case C it was commented that it is difficult to follow the directional the ASIF model. This is particularly the case when one node (e.g. a source node) relates to numerous other nodes (e.g. activity nodes triggered by that source node). It was further noted that since queuing times were not included among the activity attributes (see Section 8.2.1.2), any lead times implicitly derived from the model might be misleading.

The case study was performed in an initial face-to-face workshop, followed up by a video conference, totaling approximately five hours.

8.4.1.4 Case D

The case study was performed in two short workshop sessions, following an introductory and educational phone conference, totaling approximately two and a half hours. This is a dramatically shorter time frame than in the other cases, which is in part explained by the explicit requirement from case D line management to be as time efficient as possible. Consequently, questions and discussions of a more strategic or philosophical nature (e.g. whether change content based separation of the integration best to accomplish it), which regularly surfaced throughout all workshops, were abandoned in favor of finishing the models on time. We posit that this fact may be significant in light of the differences in interviewee responses between case D and other cases discussed in Section 8.4.3.

8.4.1.5 General Observations

In all four cases, a significant amount of time was spent introducing both the modeling techniques and the concepts underpinning them. This occurred in various forms – including introductory phone or video conference, individual study of published literature and sharing of previous examples – depending on the circumstances and wishes in each particular case. All of these methods worked satisfactorily, but the common experience from all four cases was that we were unable to dive straight into the actual modeling work without some form of miniature introduction course.

Furthermore, test coverage estimation sparked intense debate in all of the studied cases. The participants were instructed to estimate coverage as a percentage of the total space of known – explicitly or implicitly – requirements. Some provided very low estimates, even though their internal discussions leading up to that estimate were very similar to those who provided much higher estimates, implying that these values in the models are highly subjective. It is noteworthy that the attitude towards the importance of coverage varied. When giving low coverage estimates some participants would be apologetic and quick to mention plans for improvement, whereas one participant, himself responsible for some of the lowest estimates of the entire study, stated that "I think it's sufficient. We deliver this every day, and the customers are satisfied."

In all cases, the test category "edge" [Nilsson 2014] prompted questions and in some cases debate. Numerous participants thought it was vaguely defined in relation to the other categories (for instance, it could be argued that edge test cases are merely a form a functionality and/or quality test cases) and that it was difficult to determine what would count as "edge" coverage in their particular project. One participating engineer even argued that high edge coverage isn't desirable at all, since edge cases is precisely what you don't want to test for in repeated and/or automated test suites (but rather search for in exploratory testing).

Similarly, in all cases except D there were questions regarding the "new functionality" test category. In the CIViT model this is defined as testing what is "currently under development" [Nilsson 2014], but this is open for interpretation. When does something stop being "under development" and become legacy, and at what granularity is this distinction made? To exemplify, one interpretation is that something becomes legacy once it has been released, but such a definition would mean that in case B everything is "new functionality". To resolve this we agreed on a specific point in the integration as baseline for further changes – as the point where "new functionality" becomes "legacy".

After having created the CIViT model and shifting focus to the Wanted Position ASIF model, in each of the cases we noted that the participants didn't use the CIViT model directly to phrase requirements which might then be represented using the Wanted Position ASIF. Rather, while the CIViT model sparked and facilitated discussions regarding testing capabilities in general, it was complaints and improvement suggestions that the engineers already had on their minds before creating the models that tended to end up in the Wanted Position ASIF. In other words, the engineers tasked with building and maintaining the integration systems already knew what they would like to improve, and CIViT did not change those perceptions.

The improvements included in the Wanted Position ASIF models involved changes to activity attributes, new automated activities and in one case a restructuring of causal relationships between the activities.

8.4.2 Interviews

As described in Section 8.3, semi-structured interviews were carried out with a total of ten engineers participating in the multiple-case study, with representatives from all four cases. The data gathered in these interviews was analyzed alongside our observations during the modeling workshops, as described in Section 8.4.3.

8.4.3 Analysis

As described in Section 8.3, the multiple-case study data was thematically coded and grouped into five themes, corresponding to the subsections below, the first three of which address research question RQ_1 (see Section 8.2.2), while the fourth addresses research question RQ_2. The final theme, related to RQ_1, provides improvement suggestions and feedback on the studied techniques that surfaced during the modeling workshops.

8.4.3.1 Understanding and Communicating

The interviewees in their responses strongly support that continuous integration modeling is useful for both understanding continuous integration systems and communicating around them. Interviewees state e.g. that "there is a better understanding now", that "this way of modeling is useful to explain to others what you're doing" and that "in a very short time we have achieved a very good result and understanding in our organization".

With regards to the modeling experience itself we found that it was "a good tool for creating a discussion around [...] continuous integration" which led to "needed understanding and internal alignment", in the words of one interviewee. Others noted that "it was useful to do it in a group, because it opened the eyes of a number of people", and that the modeling process "helped me think" and revealed that the engineers going into the modeling sessions "had a simplified view of reality". Once created, the models were also considered helpful, however, as they provide "more of a common language", they made it "fairly easy to talk about wanted position" and serve as a basis for communication.

Only two of the interviewees, both from case D (see Section 8.4.1.4) expressed negative views in this area, but these were not directed at modeling in general: while "you need something like this" they had previously been drawing ad-hoc diagrams similar to ASIF within their team and felt that the techniques used in the study did not reveal more than they already knew. They pointed out, however, that a common model is better when communicating outside your team, or if one wants to compare or standardize practices, and their colleague argued that the studied techniques had advantages over what they had previously used, such as putting "the spotlight on the data".

8.4.3.2 Basis for Improvement

In our multiple-case study we found two ways in which continuous integration modeling may serve as basis for improvement: helping to identify and plan improvements, as well as supporting subsequent technical work.

- **Improvement Identification and Planning.** Of the ten interviewees, eight considered the use of the studied models in combination to plan improvements an effective practice, stating e.g. that "it becomes clear, the see the different stages" and that "you get a strategic picture [where it is] easy to discuss where you want to get to by putting in new boxes or changing the ones you're unhappy with, then you can bring that into a discussion about who is responsible" (in response to IQ_4, see Table 20). Seven of the interviewees were positive to the benefits of CIViT in order to e.g. "know which parts we need to look at", that it "provides greater clarity" and that "it became very clear [...] both where the boxes belong, but also that there are gaps here". One interviewee offered that it would be beneficial to make the identification of improvements more concrete by creating Wanted Position variants of both models, commenting that when "we discussed where we should go ASIF-wise I tried to connect back to CIViT, so I think I would have wanted that part first".
- **Support for Technical Work.** Six of the interviewees were positive to using modeling to support writing requirements and/or technical specifications (in response to IQ_3, see Table 20), one of them stating that they had already put it to use in this way, saying that "yes, it helps, it's helping us right now". Of the four who were not outright positive, three stated that while they couldn't see it being useful to them at present, under other circumstances or in other projects it could be (e.g. stating it would be useful in a different context or "if we worked more systematically with [these issues], but right now we don't").

8.4.3.3 Facilitation of Effective Modeling

From analysis of our experiences from conducting the case study workshops as well as of the total interview material, we propose four recommendations for effective modeling of continuous integration.

- **Multiple Occasions.** In contrast with previous studies performed by us [Ståhl 2014b], in each case of this study the modeling workshops were split into multiple occasions. At first this was out of necessity, but several of the participants remarked positively on the result, e.g. saying that "we would not have achieved the same results if we had hastened through it in a single workshop". Comparing with our earlier single workshop case studies, we agree with this sentiment. The time between sessions allows the participants to digest the new information and reflect. We believe this is of particular importance to participants who are unacquainted with the modeling techniques and require time to absorb a perspective on software integration that may be completely new to them.
- **Mixed Participant Roles.** Comparing the four cases of the study we find that the cases with a mix of different participating roles, such as developers, architects and managers, generated the most interesting discussions. This was particularly true in cases A and B, which also were the cases where interviewees most pointed out the benefits of the models in alignment and communication (e.g. "a good way to achieve alignment between the various involved parties") and tended to think that the primary value of creating the models was the workshops themselves, saying that "it's the discussion and who participates that's interesting" and "the most important thing is to have a skilled person who can lead the discussion forward and to have the right participants".
- **Coverage Versus Confidence.** As described in Section 8.4.1.5, in each of the four cases it proved extremely difficult to estimate test coverage. While we had anticipated this and used only very imprecise intervals as coverage values, this was clearly insufficient. While attempting to provide estimates the participants tended to get into lengthy discussions, some of which added value (e.g. "What does this activity do, really?"), but mostly distracted from the topic of the workshops. Instead, what we find is that practitioners are more concerned with confidence in the software: while test coverage is seen is a way to reach that confidence, in itself it is not of primary interest.
- **Complementing Levels of Abstraction.** Just as it is broadly accepted good practice to work at multiple levels of abstraction in software engineering in general, in this study we find that this also applied to continuous integration systems: the studied models were recognized by participants to serve different complementary purposes, together providing a richer view of the system.

8.4.3.4 Studied Techniques as Complements

The interviewee responses support the notion that multiple continuous integration modeling techniques can effectively serve as complements. As two engineers put it, the studied techniques are "a good combo, because they have completely different target audiences" and "A strength in CIViT is its simplicity. It's easy to phrase a message based on CIViT. ASIF is better suited to communicating with the people who will do the actual work."

In response to IQ_1 and IQ_4 (see Table 20), respectively, seven and eight of ten interviewees were positive, while in response to IQ_2 only one gave an unqualified positive response. This correlates with our observations of the workshop participants' behavior (see Section 8.4.1.5), telling us that while one model can potentially serve as input to the other, this doesn't necessarily have to be the

case. Indeed, several of the interviewees opined that closer alignment of the modeling techniques would be valuable, saying that "if you can combine them that would be a good thing" and envisioning how "if you created an ASIF model you could generate a CIViT view from that, and if you manipulated [that CIViT view] that would result in requirements on the ASIF view".

8.4.3.5 Feedback on Studied Models

Through application of the models we identified four areas of difficulty related to the modeling techniques themselves, which may serve as areas of improvement in any future evolution.

- **Multi-Faceted Activities.** The basis of the CIViT model is a two-dimensional diagram where test activities are plotted according to their level of system completeness and their feedback time (see Figure 32). During our study we repeatedly ran into situations where test activities, rather than occupying a point in that diagram, cover an area. Splitting them into multiple icons is not an appealing solution: if estimating an activity's coverage with any degree of accuracy proved difficult (see Section 8.4.1.5), estimating the coverage of the subset executing in a particular context with a particular feedback time is that much harder.
- **Test Coverage.** As previously discussed, arguably the greatest difficulty throughout the conducted workshops proved to be coverage estimations, and the discussions generated by these estimations rarely led to any tangible results. As a consequence we question the value of separating coverage types into the four categories used in the study: separation of functional and non-functional coverage is widely recognized and caused no objections, but the other categories only served to muddy the waters. Furthermore, we find that manually estimated coverage numbers are not only unreliable, but highly subjective (see Section 8.4.1.5).
- **Information Density.** Several workshop participants commented on the information density of the models, particularly in, but not limited to, the case of ASIF. One related how he with favorable results had stripped CIViT of all coverage and automation data, leaving only the test activities plotted into the diagram, when presenting to uninitiated audiences. One measure to reduce the clutter of the models introduced by us during the modeling sessions was a simplified representation of janitorial tasks in ASIF (see Figure 33). At the same time interviewees were largely positive to ASIF, e.g. as a support for technical work (see Section 8.4.3.2), so clearly there is value both in simplified visualization and in information richness. Consequently we believe that there is room for improvement for both CIViT and ASIF in this area.
- **Manual Versus Automated Creation.** There is considerable manual effort involved in creating the models of this study. The total time spent modeling ranges from approximately 9 to 30 man hours, not counting the time spent by the researchers conducting the workshops. However, much of the time was spent on the type of discussions that many of the participants considered the most prominent benefit of the workshops. It may also be time well spent in order for the participants to become acquainted with and appreciate the modeling techniques – in the words of one of the participants, "When I first read about this I felt [very skeptical], but after the [first sessions] I began to understand what the value was [and] then grew increasingly positive as time went on". Our conclusion is that in the ideal case the modeling techniques would offer two creation processes: one manual to encourage debate and alignment, and one automatic that simply generates the models quickly and cheaply.

8.5 Related Work

There is an increasing amount of published continuous integration research available, and much of it falls into two main categories: experience reports and presentation of a tool, framework or technique to facilitate continuous integration. In previous work [Ståhl 2014a] we have conducted a systematic review of literature with "continuous integration" and "software" in their titles or abstracts, where 76 publications were fully reviewed in search of descriptions of continuous integration practices. Several papers successfully identify challenges we have encountered in our own work, e.g. scaling continuous integration [Roberts 2004, Rogers 2004] and communication [Downs 2010], but to our great surprise, it's very difficult to find anything relating to continuous integration modeling or, in a broader sense, the ability to reliably and systematically design conducive and holistic continuous integration systems from end to end.

Revisiting the issue in the work reported from in this paper we conducted another literature review, searching specifically for modeling and visualization techniques, using the inclusion and exclusion criteria shown in Table 21. This yielded a total of 60 results. Application of exclusion criteria EC_1, EC_2 and EC_3 removed 55 results from the set. A number of these results appear relevant at a glance, e.g. dealing with "incorporating [visualization] tools into the process of continuous integration" [Bartoszuk 2014], applying continuous integration to model-driven software development [Diaz 2012] or including model based testing in continuous integration [Mossige 2014]. These topics are, however, distinctly different from visualizing or modeling the continuous integration system itself. Of the remaining five results, one discusses the automatic configuration of developer workspaces [Rouille 2013] and one is a brief description of a tool that mines unit test results and coverage data from e.g. XML files to produce dashboard metrics [Remencius 2009]. The final three results are papers discussing ASIF or CIViT, in which we have participated.

Inclusion Criteria	
IC_1	Any published material matching the Scopus search string `TITLE-ABS-KEY("continuous integration" AND ( model OR modeling OR visualization OR visualisation )) AND SUBJAREA(COMP)`.
Exclusion Criteria	
EC_1	Material not available to us in English or Swedish.
EC_2	Posters for industry talks lacking content beyond abstract and/or references.
EC_3	Material found through review of abstracts not to present any techniques for modeling or visualizing continuous integration systems.

Table 21: Inclusion and exclusion criteria of a literature review of continuous integration modeling and/or visualization.

Reasoning that since a significant portion of the work done in a continuous integration system typically revolves around testing, there may be relevant techniques for planning or orchestration of tests within a continuous integration system. Consequently, we conducted a third review, using the same exclusion criteria as before, but now using a different search string: `TITLE-ABS-KEY("continuous integration" AND test AND (plan OR planning OR`

`strategy OR orchestration OR scheduling)) AND SUBJAREA(COMP).` This search yielded 17 results, of which 14 were eliminated by the exclusion criteria (see Table 21). The remaining three papers were reviewed in full. Of these, one describes an extension of the FitNesse tool to manage a number of automated testing frameworks [Kim 2009a], another similarly presents a framework that is a "streamlined pipeline [that] will be effectively operated in all phases of the software development life cycle to evaluate the future of the software product", but offers little in terms of deciding which tests to run when, why, where and how [Rathod 2015]. The third provides a thorough assessment of test case prioritization methods, motivated by the observation that in a continuous integration scenario, "the time available for regression testing [...] is always limited" [Jiang 2012]. We consider this to be highly interesting work, as time constraints are a crucial factor, and indeed a driving force, in continuous integration modeling – after all, if one has the luxury of executing every single test for every single change without time constraints, then there is arguably little need to model or design one's continuous integration system in the first place. Consequently, we consider the area of test selection and/or prioritization as complementary to continuous integration modeling: each individual activity in a continuous integration system may be strictly time boxed, and within each of those activities test case prioritization may well be used to optimize the value derived from it. Unfortunately, it does not contribute to the holistic design or comprehension of the system itself.

We find this lack of published material regrettable, and in the context of the positive feedback received from industry professionals in previous work, a clear sign of a research field with great untapped potential.

8.6 Threats to Validity

External validity is threatened by the fact that the data was obtained from the application of two modeling techniques, while some of the conclusions from that data pertains to continuous integration modeling in general. We argue, however, that while generalizations to any and all modeling techniques would be problematic, generalizations to modeling practice as such, of which the studied techniques are examples, are valid.

Researcher bias is also a concern, since we have previous involvement with the studied techniques. However, the techniques were not chosen because of this involvement, but rather despite it, as there is demonstrably a lack of alternatives (see Section 8.5). Second, a certain degree of protection is offered by the research design, which is based on the direct and presumably unbiased feedback from software professionals in multiple companies and settings as the primary data source.

Furthermore, threats to internal validity include mortality and selection [Cook 1979]. Mortality, because three workshop participants could not be included in the interviews (see Section 8.3). We do not consider this to be a significant threat, as they represent a small minority and were prevented from full participation due to e.g. illness, which is unrelated to the study and can be considered random. Selection, because the workshop participants were not randomly selected, but instead purposively sampled to achieve the best possible competence profile, as our main concern was to ensure that the engineers with the required insight and experience were present (as discussed in Section 8.3). To protect against researcher bias, each case contact person was asked to provide the names. This caused differences in participant compositions, which may have affected the outcomes of workshops and interviews (as discussed in Section 8.4), but not to a degree that we consider a threat to the conclusions.

8.7 Conclusion

Understanding, communicating and architecting large and complex continuous integration systems is non-trivial. Consequently, especially considering the significant time, money and resources being invested in these capabilities in the industry, any aid that can be offered practitioners in order to build an overview of and to analyze their systems, as well as to investigate the implications of potential improvements, is a valuable contribution to the field of research. Consequently, based on previous work and the supposition that separate modeling techniques, offering distinctly different perspectives on continuous integration, should favorably complement each other when applied to the same case, we have through a multiple-case study of four large industry projects and subsequent interviews with professionals of multiple roles sought to answer the following research questions:

- **How may continuous integration modeling be applied to benefit industry practitioners in their day-to-day work?** We have found that modeling of continuous integration systems can be beneficial to industry professionals by improving understanding and communication of complicated continuous integration system, as well as for identifying and planning improvements to those system and, to a lesser extent, as direct input and support to the subsequent technical work to implement those improvements.
- **How can the continuous integration specific modeling techniques of ASIF and CIViT be effectively applied in tandem to complement each other?** We have found that the two continuous integration modeling techniques of Automated Software Integration Flows (ASIF) and Continuous Integration Visualization Technique (CIViT) can favorably be used to complement each other by leveraging their respective focus on low and high levels of abstraction. However, we have also found that there is room for improvement both in their alignment – allowing models of one technique to serve as input to the other more effectively – and in the design of each individual technique itself, as well as in improved tool support.

Finally, the study has demonstrated that continuous integration modeling techniques are applicable to a range of large scale projects in varying industry contexts: all modeling sessions could be concluded without significant difficulties.

8.7.1 Further Work

We consider both the unification of the studied modeling techniques and automated tool support for building those models to be promising opportunities for further work.

Acknowledgments

The authors would like to extend their sincere gratitude to all of the participant companies and their employees for their time, patience and insights.